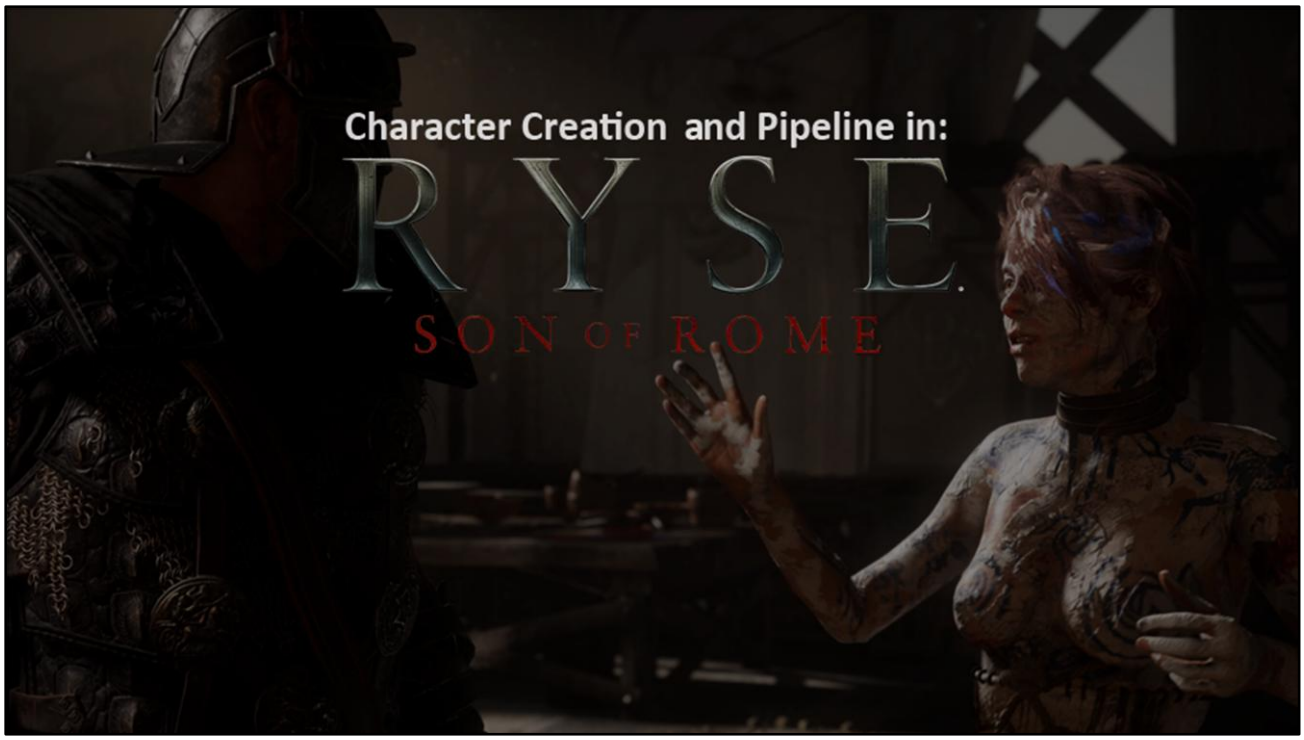

Character Creation and Pipeline in:
RYSE
SON OF ROME


Currently Lead

Senior Character Artist during Ryse production.


In total 10 in-house character artists contributed to the project at various points of the production. In addition to that we worked with14 hand picked freelancers.

At any given time through out the production the maximum amount of character artists working on Ryse was eight.

# Presentation Overview

- Concept
- Modeling
- Texturing
- Ryse Hair
- In-Game Cloth
- Facial Art Pipeline

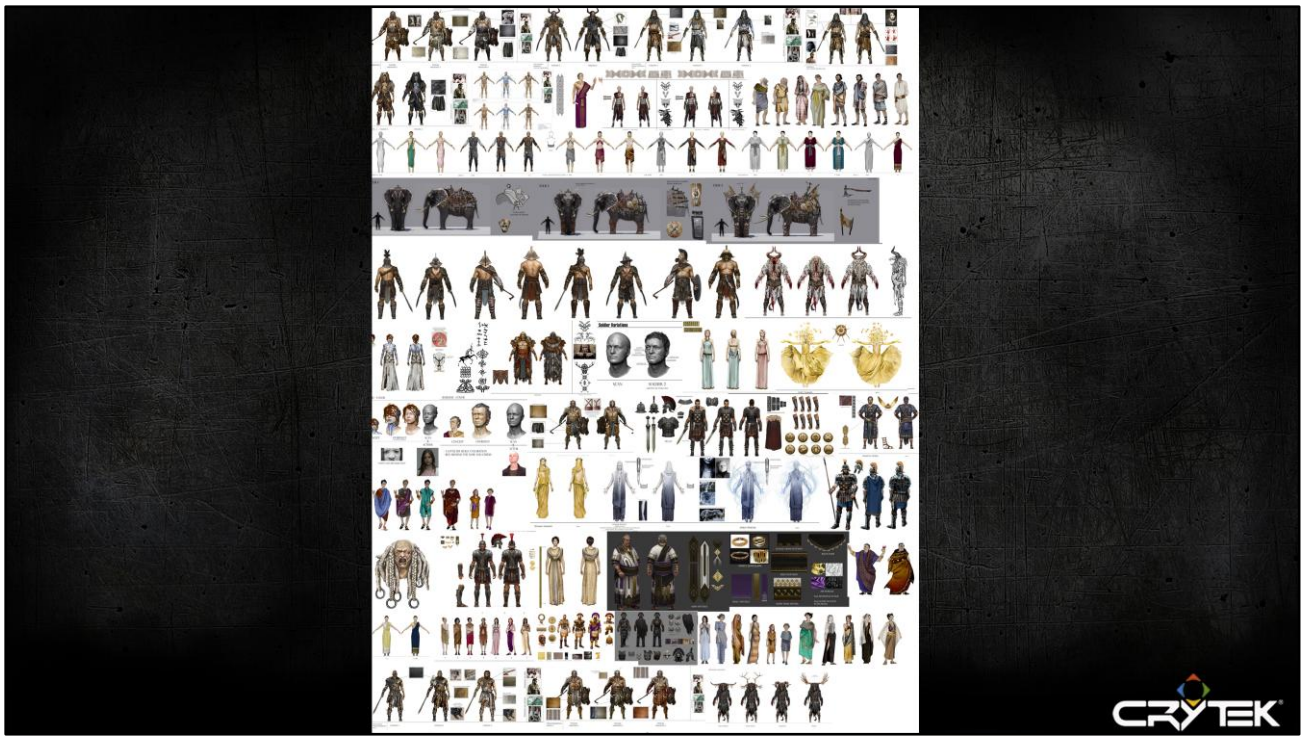

First step of bringing a character to life is the concepting phase.

The concept artist will work close to art direction and gameplay.

The character artist will keep an eye on the iterations of the concept, to get an heads up on what technically might be needed later on (hair, cloth).
The artist will also consult the concepter if certain aspects of a design wouldn't work well with movement or deformation.


Here is an example of a story characters concept sheet.

Contains break down of certain important parts (the harness, helmet), also material references from photos that the concept artist intends the final asset to have.

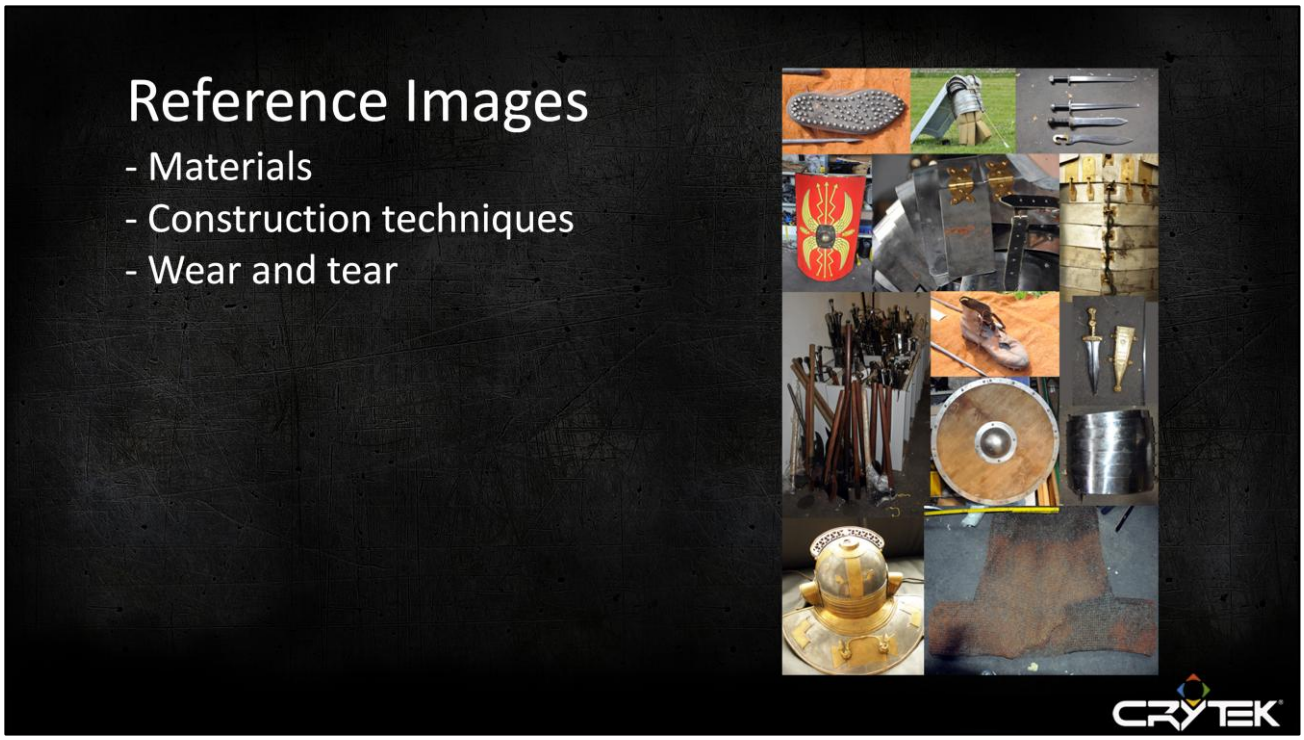



As you can see a lot of concept art was created, and these are only the final approved ones.

There were a lot more iterations and variations of every single character seen here.

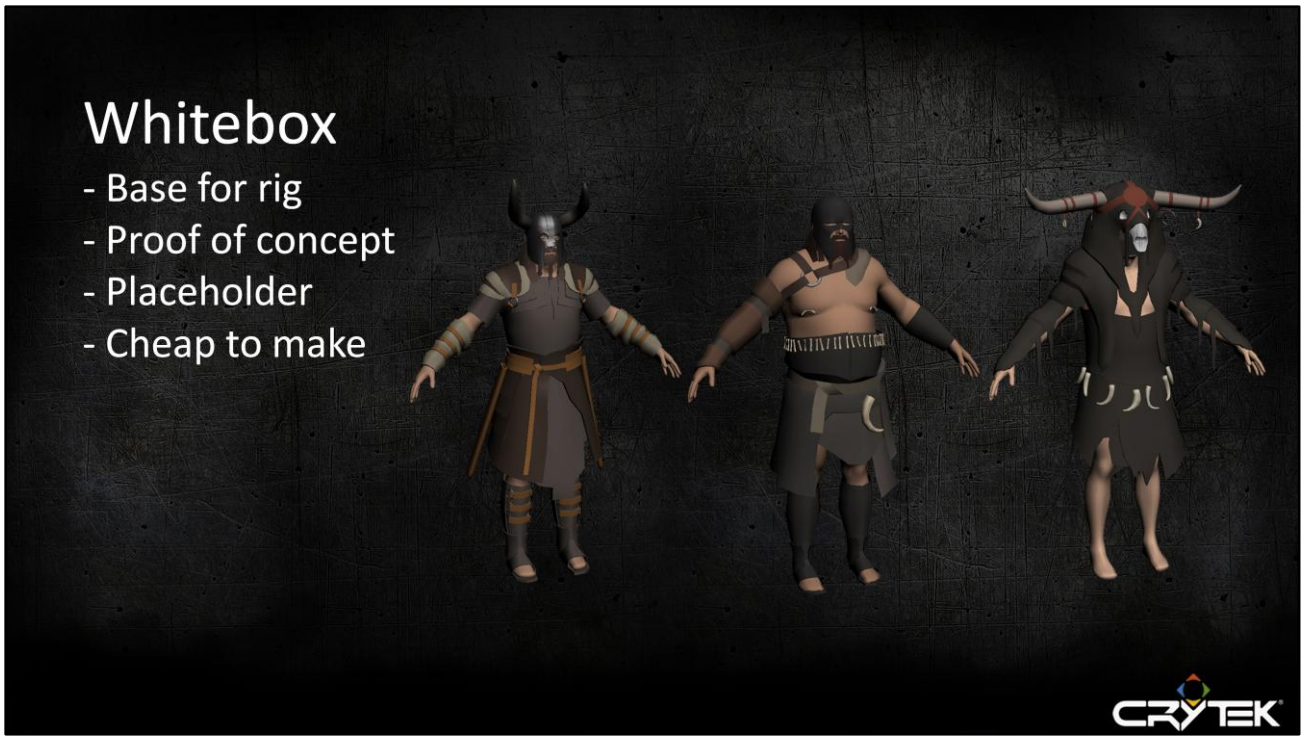

As a rule of thumb, the more important a character a character is in the game the more iterating goes in to the design.

Marius the main character being a good example of this.




Once the concept is approved we would gather real life references to get a better understanding on how things were built and what materials was used in real life.

We ended up with a huge database with mostly pictures from the internet, the ones on this slide was taken by team members during a field trip.



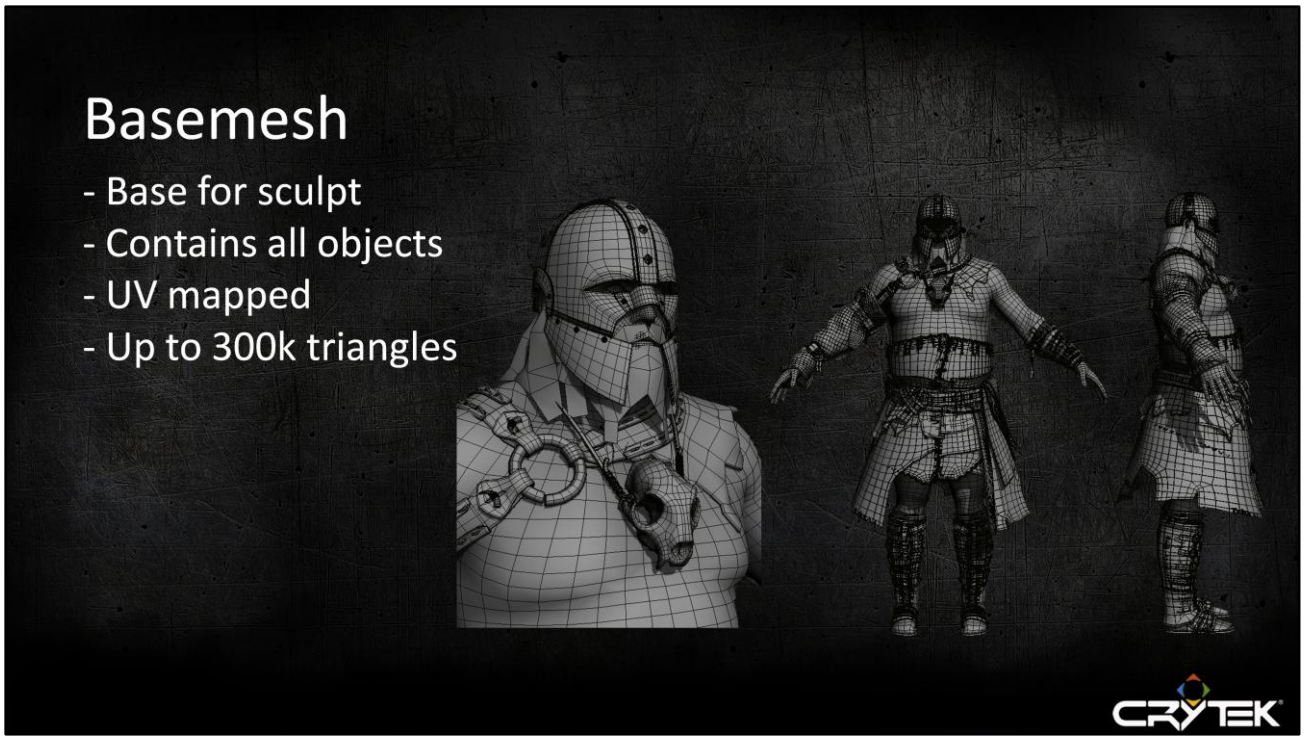

The first step of creating the asset is the modeling of the whitebox.

Because the whitebox will be the starting point for the rig all the parts that will be animated or physicalized should be represented. This includes smaller objects attached to jiggle bones and such.

The rigged character is then sent out to be tested by various departments (animation, gameplay), and iterated upon if needed.

The whitebox is not actually white, to help the lighting artists we give their material correct PBR values.

Since the whitebox is merely a placeholder as little time as possible should be spent on it, since it will be thrown away in the end.




The basemesh is the beginning of creating the actual final asset.

It will later be exported to Zbrush, subdivided and sculpted upon, because of this an even polygon distribution is important to keep in mind.

We also UV mapped the basemesh, since in this project we textured the high polygon version of the model. More on that later during the texturing part of this presentation.

Usually this model was not sent to other departments for testing, but there were exception as you will see in the coming slides.

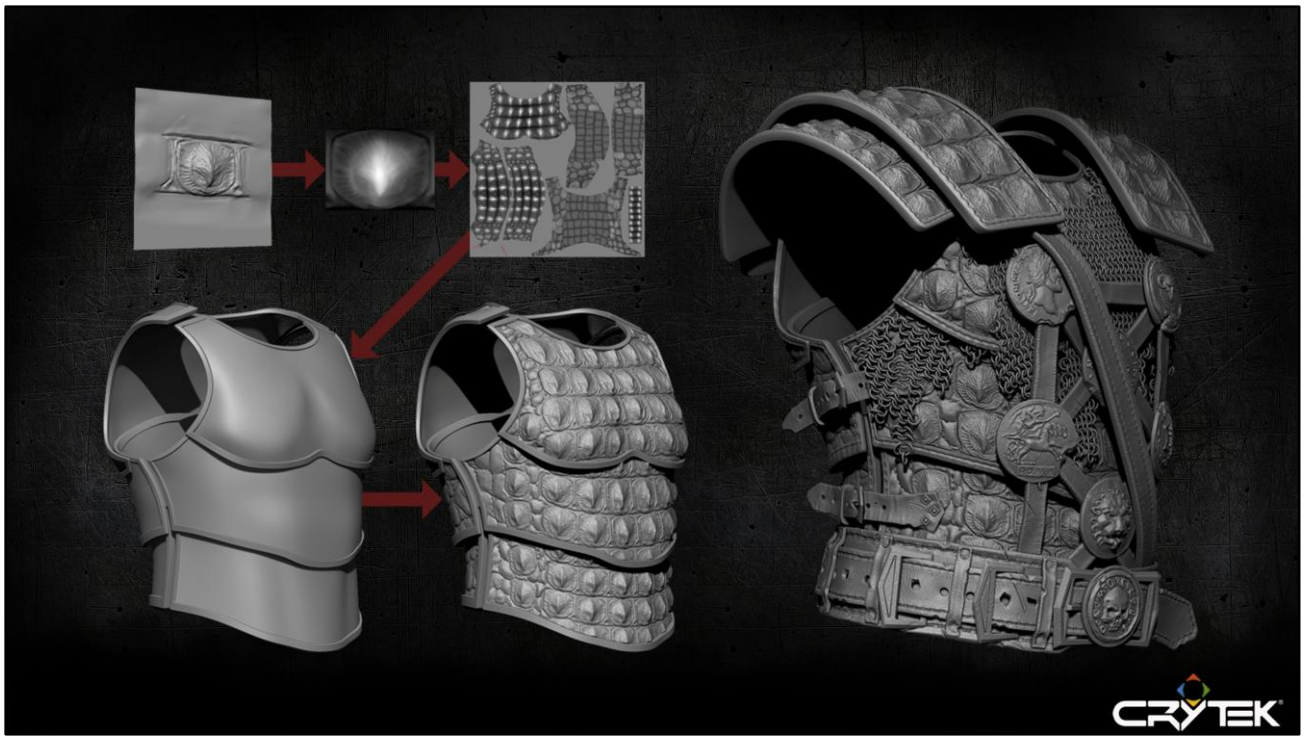

In this case the praetorian basemesh was used for testing materials on by the shader programmers.


Basemeshes would also sometimes be used for reviewing hero assets, keeping the directors confident things were heading in the right direction.


The sculpting techniques varied between artists, and no strict way of working was necessary.

In general the artist tended to work in a very non destructive way using layers. Procedural methods was used to generate small details quickly, like the surface texture of cloth.

Personally I enjoyed taking advantage of the high poly model being UV mapped, giving one more way to get information in to the mesh.




Like in the case of Damocles crocodile harness. Were I started out with sculpting a single scale.

A displacement map was baked from the scale. In Photoshop I copied and tiled it to later morph the pattern with the liquefy tool.

Then I would bring the map in to Zbrush and displace the mesh, I could quickly jump between the software's iterating the design really fast.

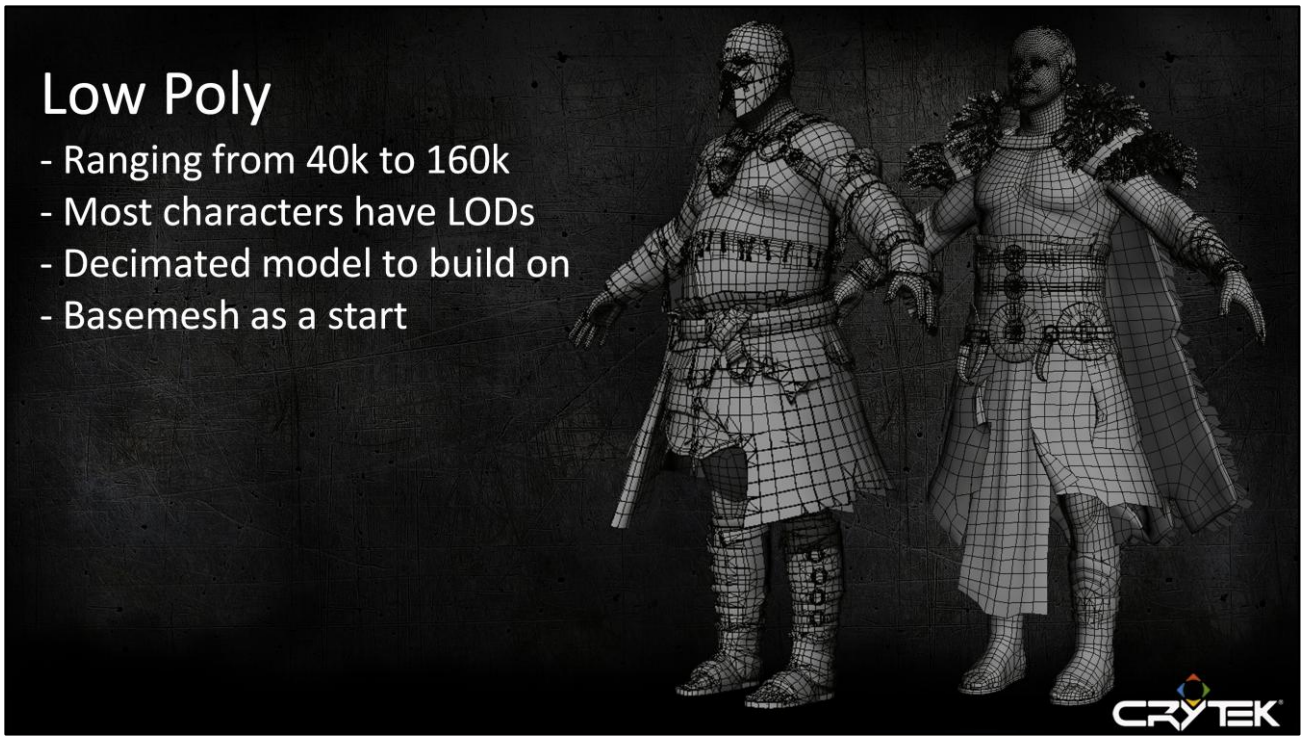
Later a quick sculpting pass was done, techniques like these saved a lot of time, so more time and attention could be spent sculpting other parts.


A few more angles of the details.

The chainmail was simulated using MassFX (similar to ncloth I believe) in 3Ds Max.


As you can see on Marius armor every single scratch and dent was reprecented in the high poly sculpt.

And in some cassses the final amount of polygons we would end up with on a character could reach 130 million.
This made it necessary to split the asset into multiple Zbrush files, each and every one with plenty of sub-objects.

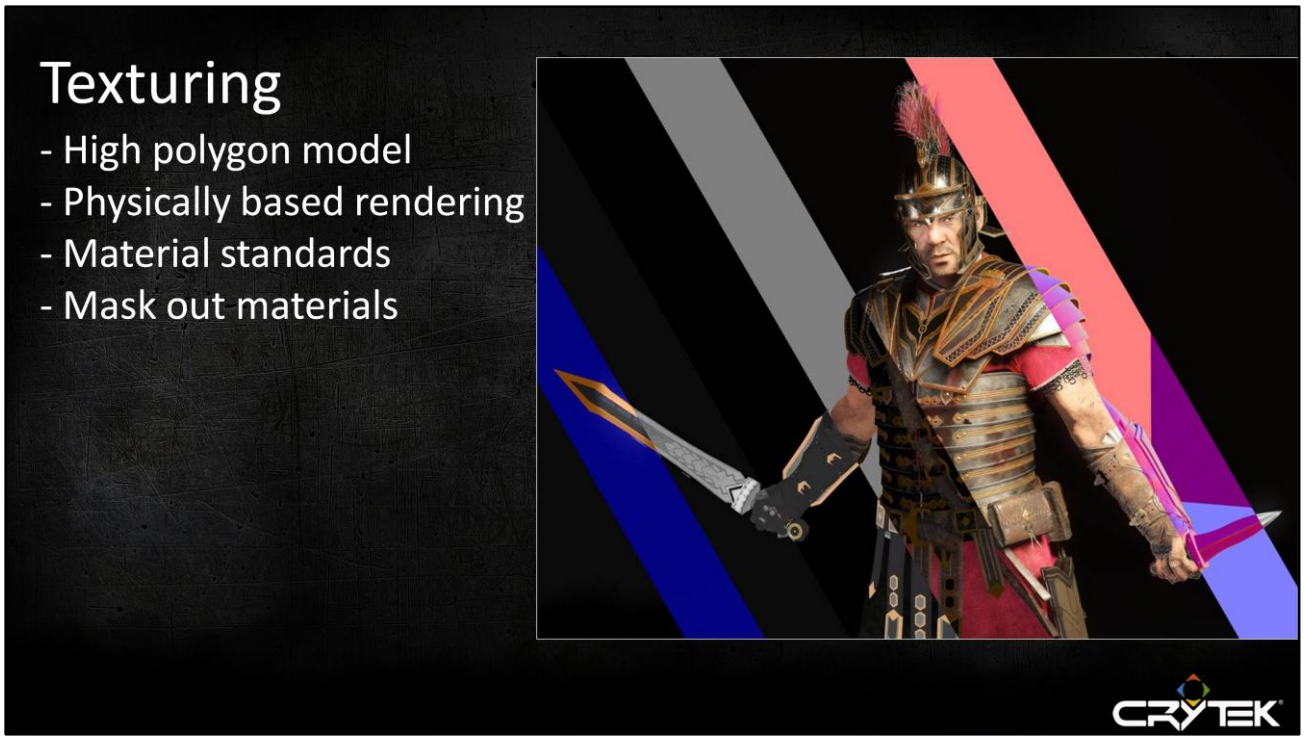



The amount of polygons would vary a lot depending how the asset would be used, Oswald here being a cinematic character had in the end around 140k.
The barbarian enemy had around 45k since he would appear in gameplay situations, sometimes with groups of enemies.

LODs was handled very much in the same way, most character had them, cinematic ones being the exception. Marius and Damocles didn't have LODs either being playable hero character and always fairly close to the camera.


Here you can see all the stages of the modeling process, from concept to finished in-game low poly.

Now lets take a look in to the texturing procedures we used.



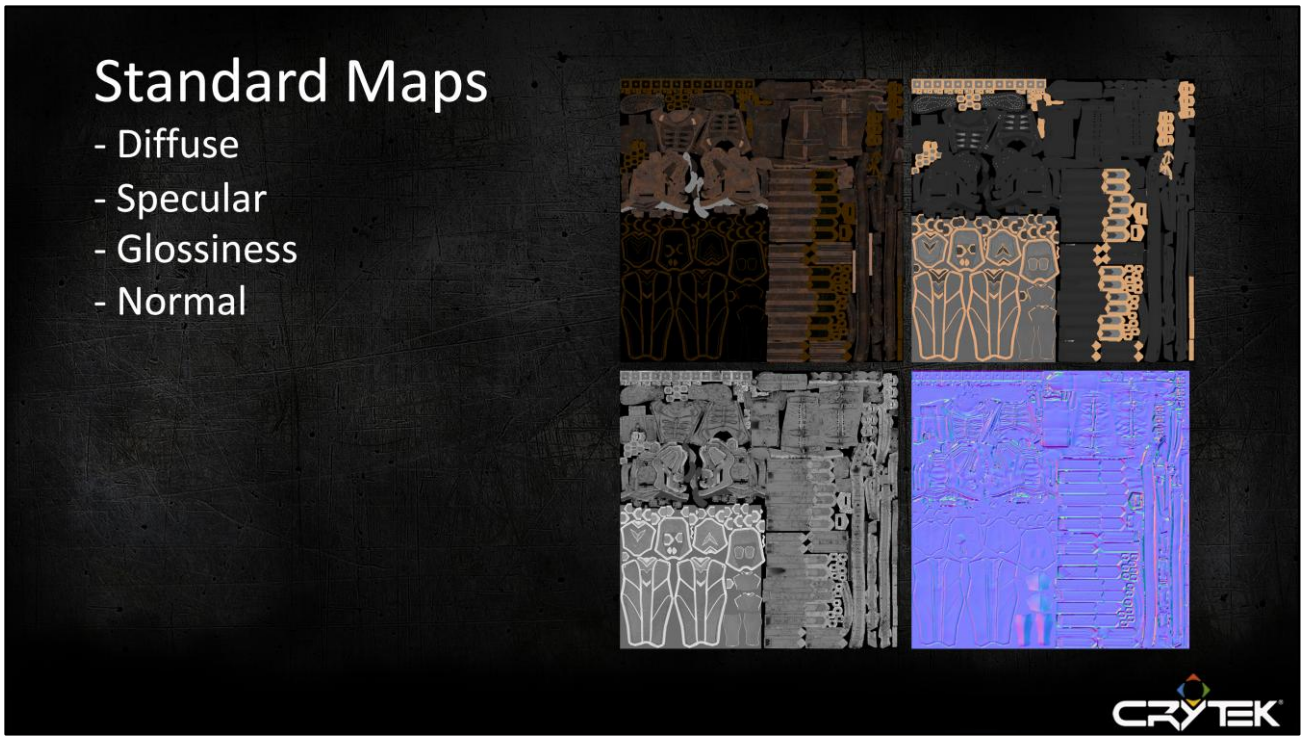

One of the great advantages of texturing the high polygon version of the character is the we could be much more flexible with the in-game version.

Since we were developing a  launch title the hardware was unknown and polygon budgets would change. Previous when we textured the in-game version of the model changing the topology or UV layout would mean we lost all the work we put into the textures, not in this case. Also LODs could get custom baked textures instead of using LOD0s, giving us better quality and fewer draw calls.

The drawback was that we had to UV map the asset twice, both low and high.

I found working with PBR being a blessing, now with set specular values it was much easier to keep concistancy beween assets and more importantly different artists. As a bonus metrials looked great in all types of light conditions, that wasent the case in previus projects.

In Photoshop the different materials had their own folder and was masked out, making it easy to iterate on.




With PBR and with the better light, shadows and in-game occlusion the diffuse maps are more or less albedo maps, the only small exception were that sometimes a little bit of occlusion would be backed in to the texture to bring out details that are so small they are not represented by the geometry at all, like the fur on a barbarians bracelet.

The specular map is more or less flat, the values are physically based so we don't tamper with them.

The  glossiness map is more or less were the magic happens. We would put in quite a lot of contrast in it, controlling rough areas, polished, adding dirt and scratches. I would say this is often the most important map.

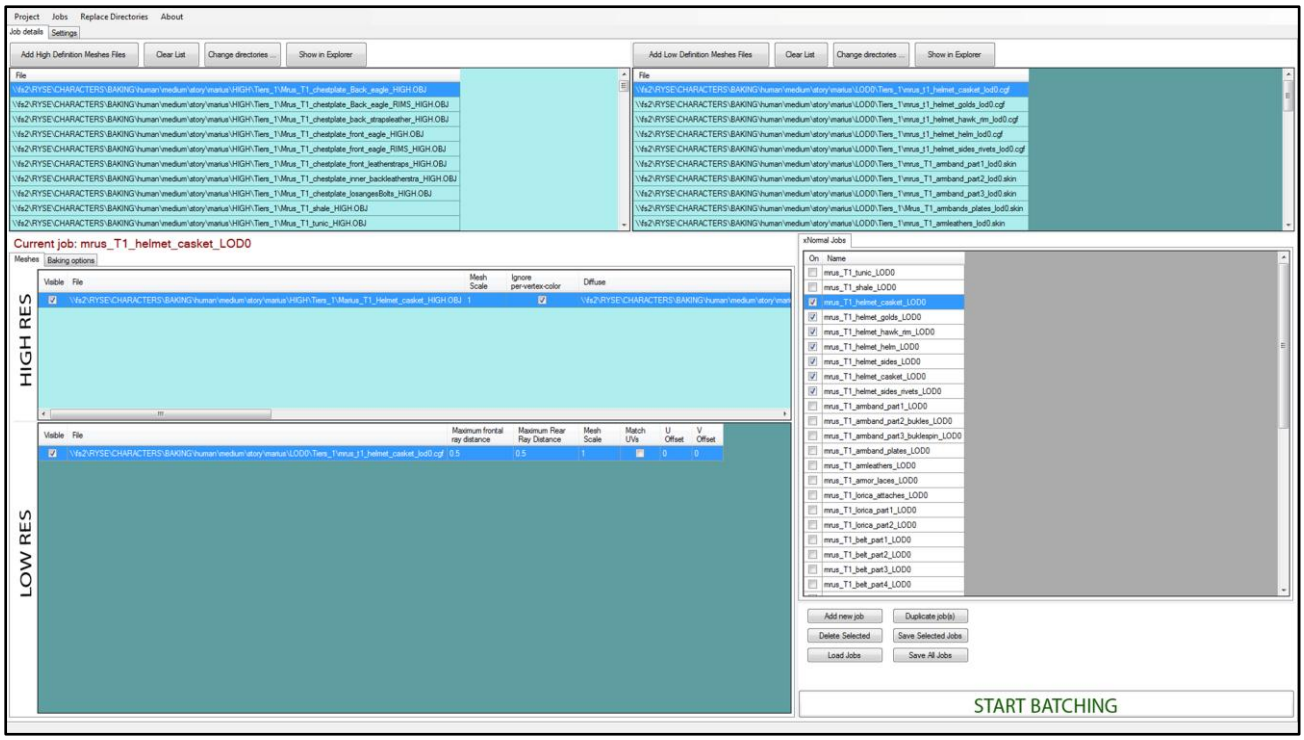
The normal maps were pretty standard so not much new to say.


As you can see Marius ended up having quite a lot of textures assigned to him, in total 32 maps for the body.

And there's even more once we take a look at the head, adding mouth, eyes and hair.

Like I mentioned we textured the high polygon model, so all of these maps needed to be transferred to the low polygon mesh.

We used to bake all maps individually by hand using xNormals, this simply wouldn't work with this amount of data.



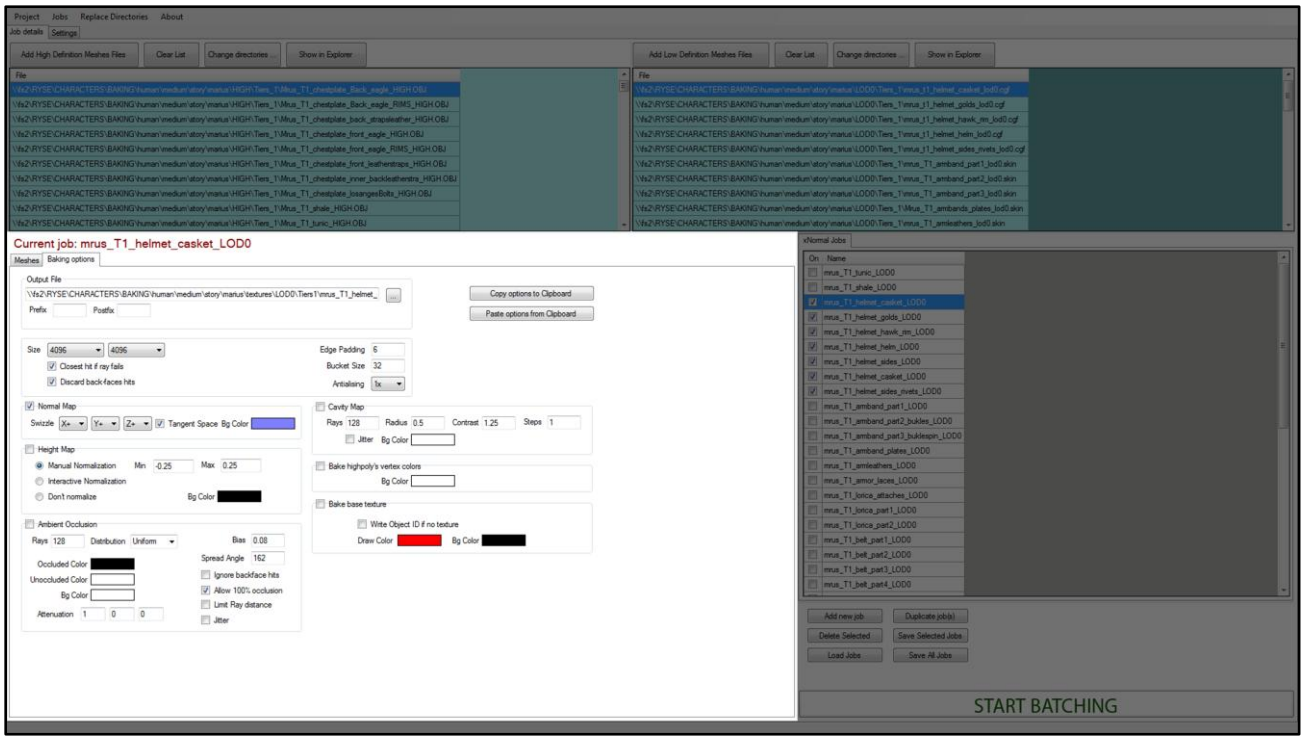

To tackle this challenge xnJobs was developed, a batch baking and organizing tool made by Bogdan Coroi.

As a norm one project file from Xnjobs would contain one characters baking ,

In the example here you can see how the legionary version of Marius was set up. In the top row all his high and low polygon assets is loaded.

Lower left we would put the particular meshes we want to bake for this job, in this case its Marius helmet.

Lower right you will see all jobs set up for Marius, we would often create many jobs instead of exploding meshes, saving a bit of time.



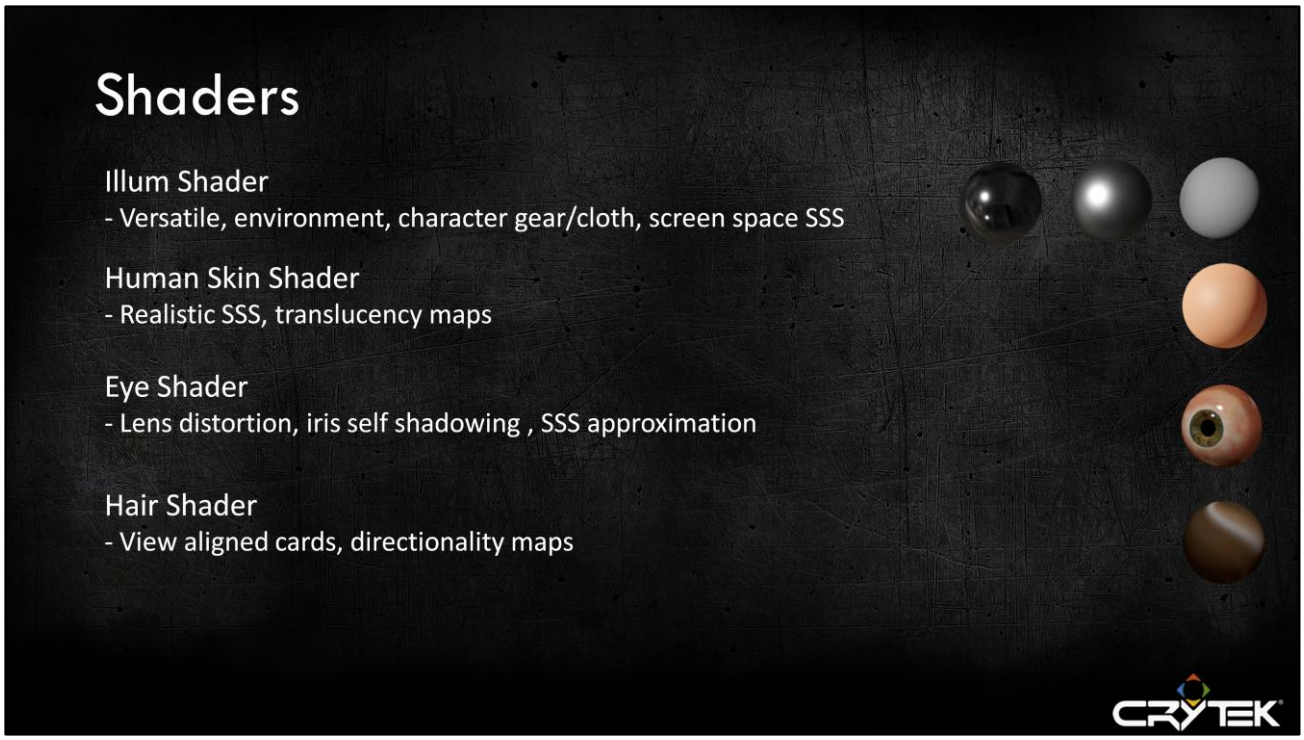

In the last window we chose if its textures, normals or occlusion we want to bake down, its not by chance the UI mimics xNormals.
Once you hit batch xnJobs actually sends all the jobs one after the other to be executed by xNormals.


XnJobs was crucial for our texturing pipeline, but we still want more.

Like I said the character were divided into different project files, so for the future we should have a batch batch baker.

The texture files on this slide comes from one characters baking alone, all these files needs to be converted to our texture file format crytif, and sometimes composited together in Photoshop. Way to time consuming and repetitive work. So that's something we want to improve.

We have also been looking into Mari lately, there's some cool features for us game developers coming. We also have a CRYENGINE shader up and running in Mari.



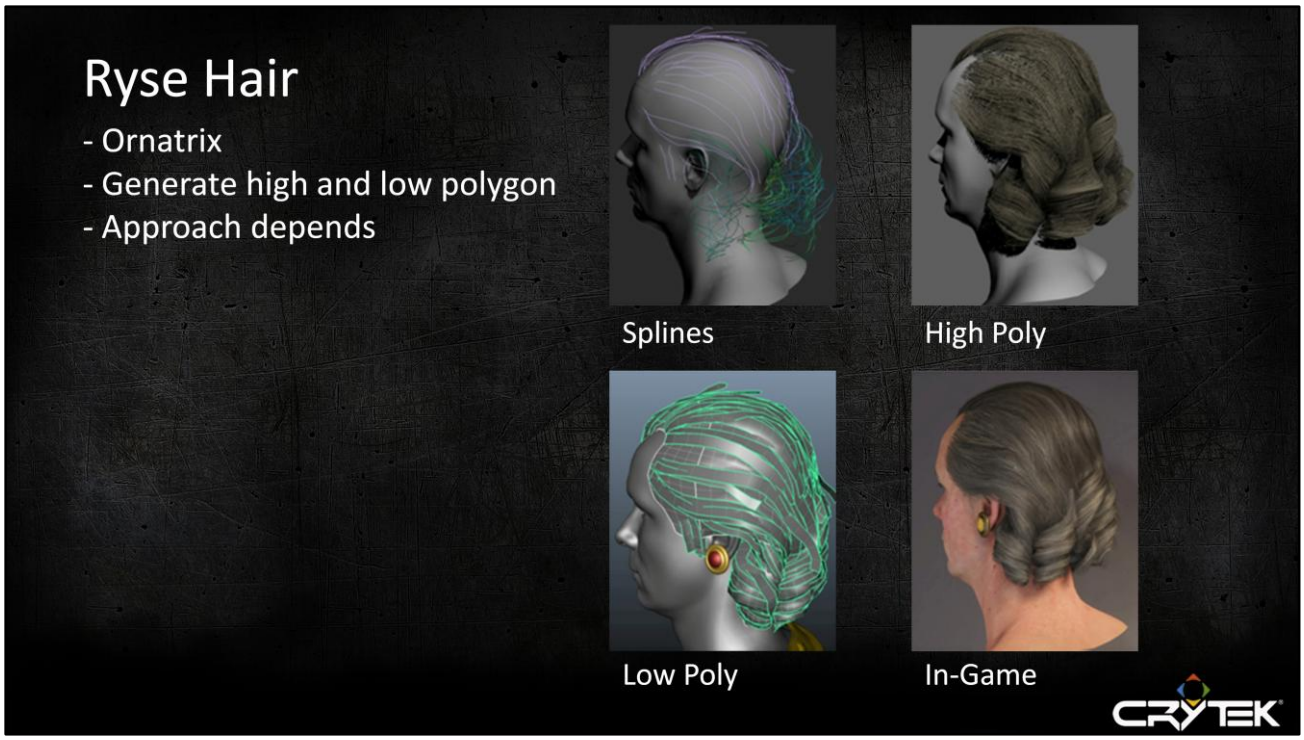

Illum shader is the most versatile and commonly used shader by our artists, environment and character alike. Most materials can be represented with this shader, chrome, wood, stone, dirt, cloth, leather, iron, the list goes on. Has screen space subsurface scattering as well.

As the name implies human skin shader is used for skin, has a very realistic SSS and a translucency map can be added to give light bleeding effects.

Since our in-game eye meshes are spheres its up to the shader has to make the eye look anatomically correct, pushing the Iris in, adding lens distortion, self shadowing and SSS approximation.

There's two new cool features we had for Ryse when it comes to hair, besides from just better alpha sorting.
View aligned cards is hair planes that will rotate to face the camera at all times, giving the illusion of much thicker hair. But you have to know when to use them, in some cases they are awesome and in some hair styles they will end up twisting and not looking so good.

Directionality maps for hair will be explained in the next couple of slides



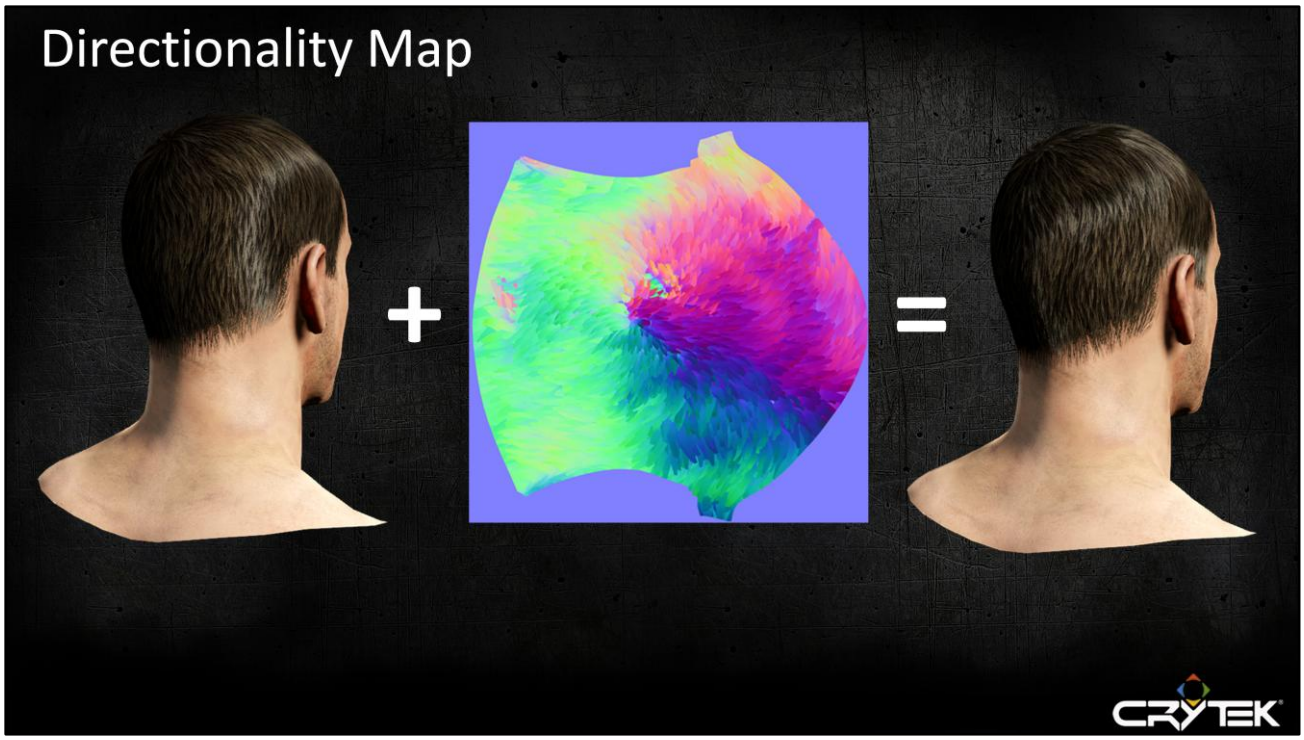

When working on hair for Ryse we relied heavily on the Max hair plugin Ornatrix. There is many similar solutions out there but we found this one suited us the best, we enjoyed the non-destructive workflow and stack based approach.

We would first use the Ornatrix hair splines to generate a high polygon version of the hair style, in the beginning this is mostly to check what looks good on the character. But later these splines will be used for generating the low polygon hair planes and the directionality map which I will talk about a bit later.

The approach the different artist used varied, one reason being personal preferences, but mostly since every hair style needs a different approach. We were all using the same tools but in different ways.


Final hair style to the left.

To the right the base hair has been removed. As one can see the base hair is crucial for the look, the next slide will go in to more detail of some of the hair base tech.

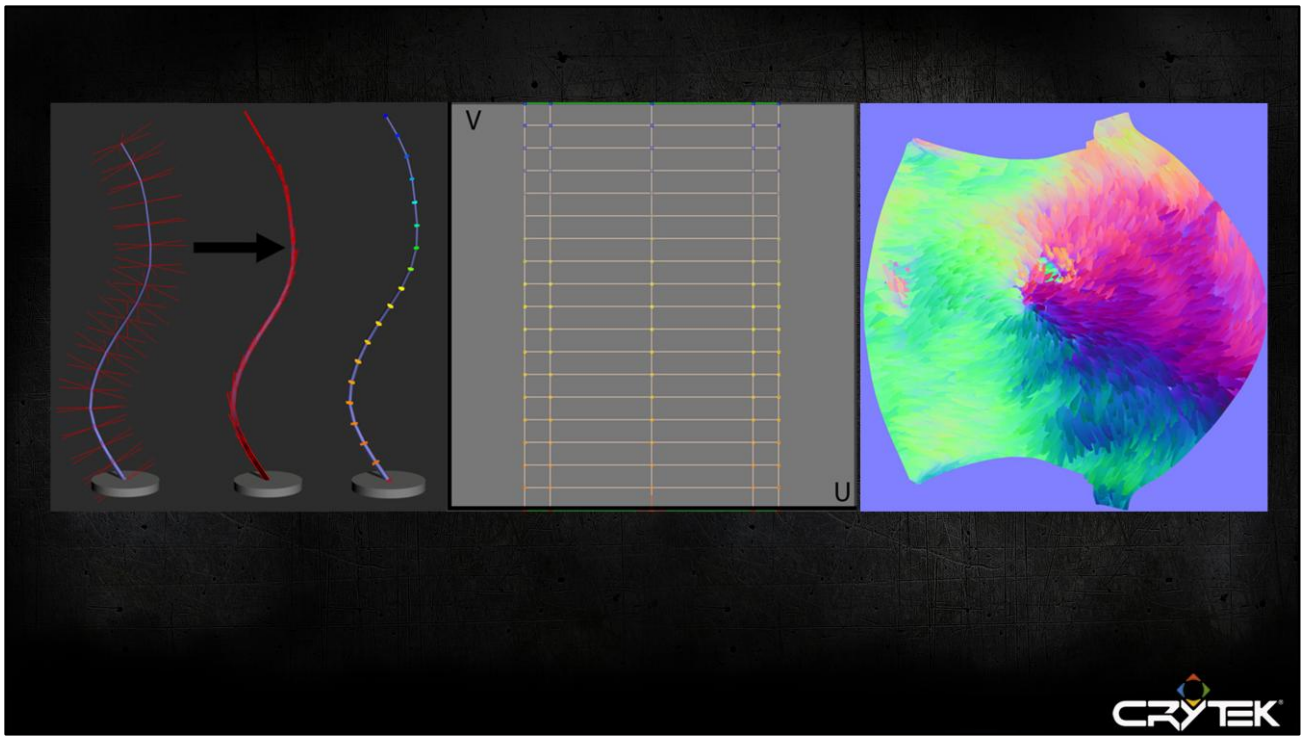



As you can see, by default the hair caps specular highlight doesn't look as one would expect, forming a strange halo shape. This is because the shader doesn't know which direction the hair is flowing.

A directionality map gives the shader the information needed to control the highlight.

The final result is an anisotropic highlight as you would expect from real hair.

Next slide will show you the approach we used to generate these maps.

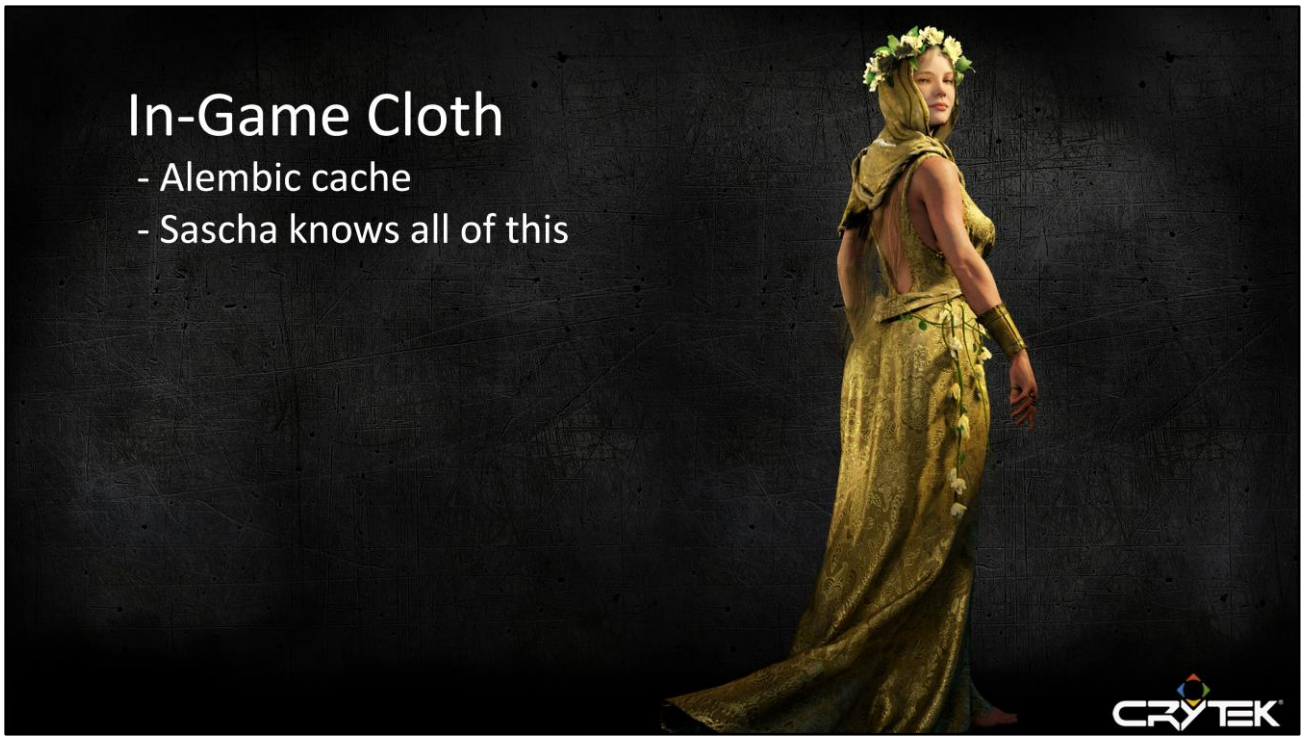



Basically what we wanted to begin with was the hair strains normals to be flowing in the direction of the hair. So we needed to rotate them all 90 degrees toward the tip. But that's where the problem is, how would the plugin we were developing know what's the root and the tip of the hair geometry.

Luckily Ornatrixs default UV layout on generated meshes has the roots UVs at V 0 and the tip at V 1. With this information the in-house plugin could realign to flow with the hair.

So when  we ran the plugin the whole hairstyles geometry would get custom normals. The practical thing is that all we had to do now was to bake a normal map from the hair mesh and the output would be this directionality map.


Final hair




There was two types of cloth simulations used on Ryse. One of them being pre-simulated and used by cinematic character.

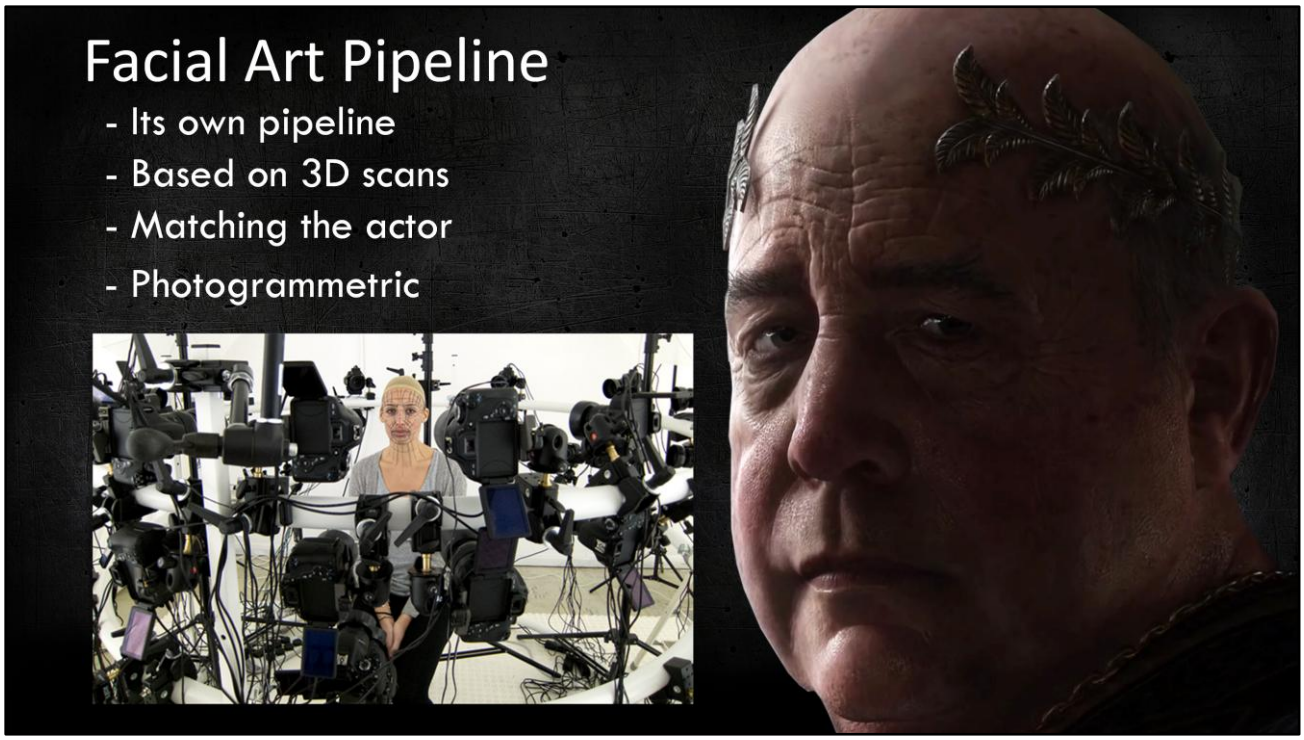
Have a look at Sascha Herforts presentation for more details on pre-simulated cashed cloth.


We also had real time cloth, developed from the ground up using feature film methodologies.

The approach is similar to a nCloth pipeline, with two meshes being used. A physics mesh and a render mesh.

The phys mesh is fairly low poly, since every vertices will make the simulation heavier, but also more refined so it is a balance. The phys mesh should also have an even polygon size and distribution for best results.

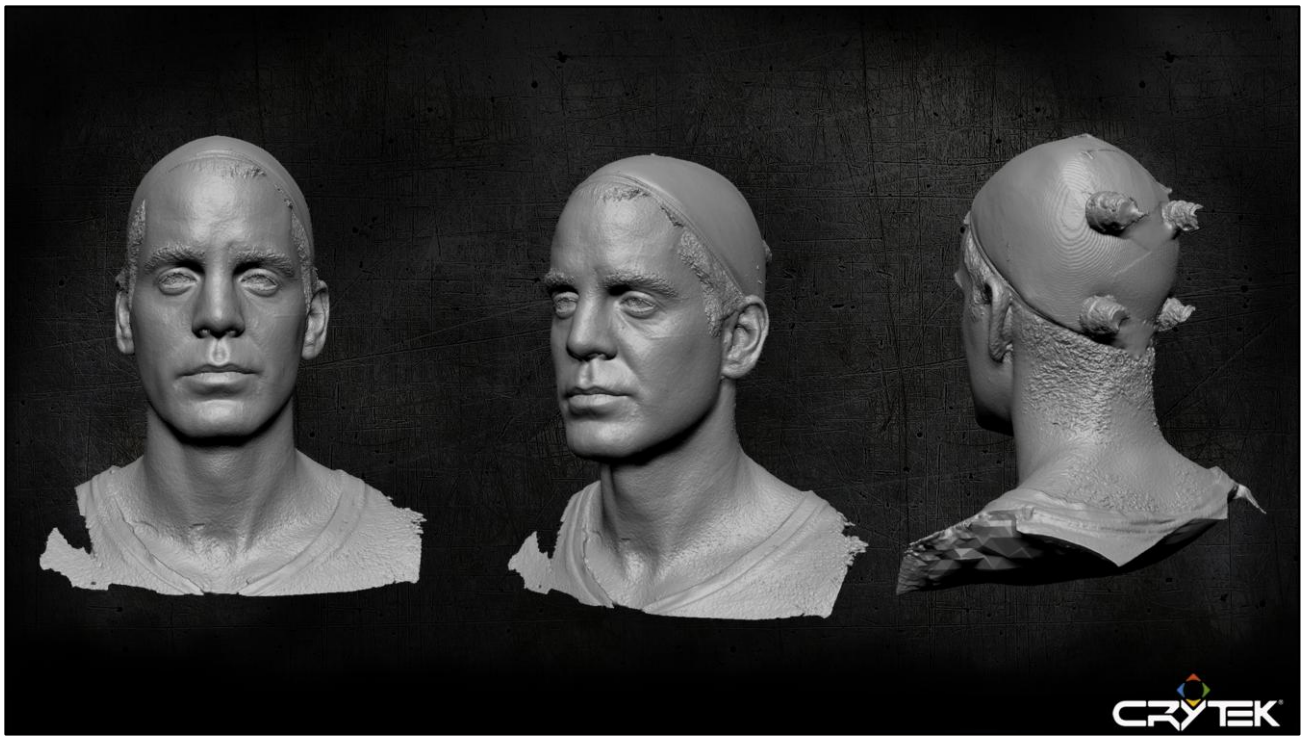
The render mesh is what the player sees in-game and can have a higher polygon resolution. The phys mesh drives the render mesh much like a wrap deformer does.




We treated the heads more or less as different assets from the bodies, since they have their own pipeline. We still tried to let the artists make both the body and the head of the same character, promoting ownership.

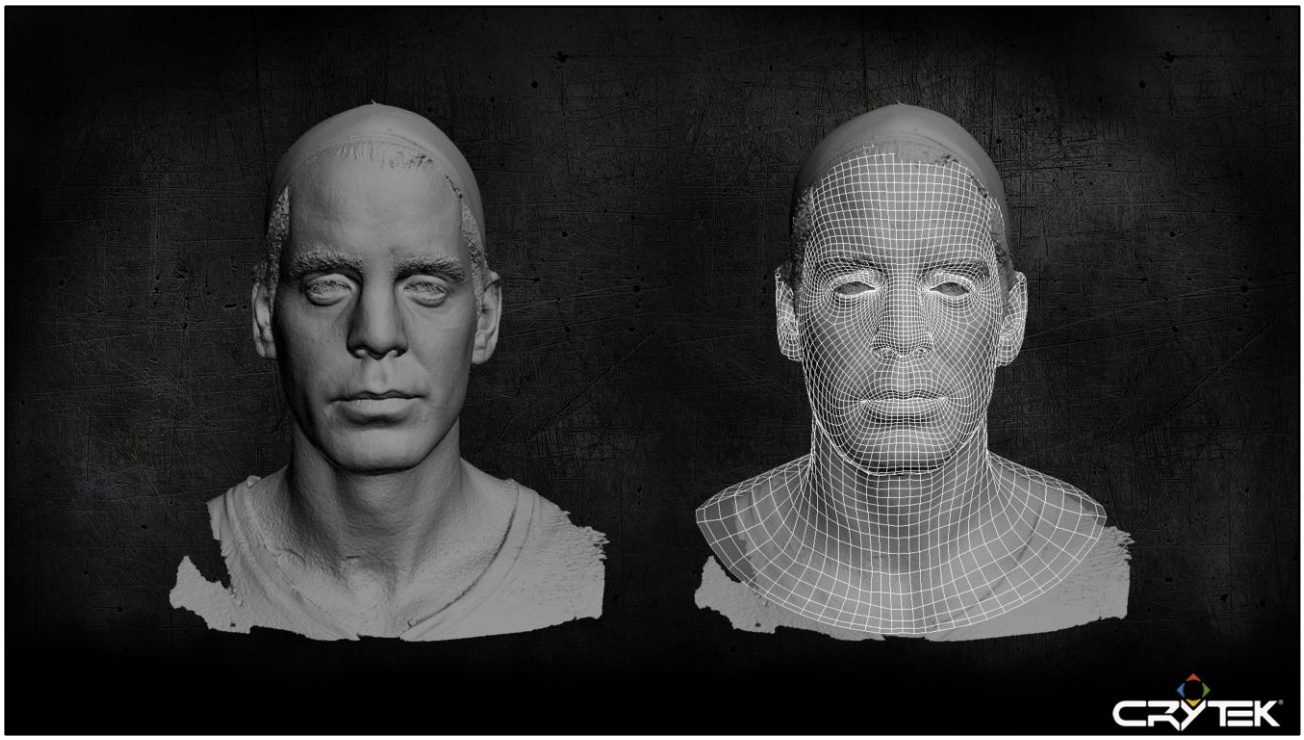
Our character heads where all based on 3D scans of the actual actors so that the performance capture data would work well.

We used a photogrammetric approach as the picture in the slide shows. Several cameras taking an image at the same time from different angles, then later being loaded into a software that turns it in to a 3D object.




Here you can see the quality of the scans, fairly good quality but still a fair amount of artefacts, and a hair cap and t-shirt etc.

So the scan data needs to be cleaned up, but first be given a good topology



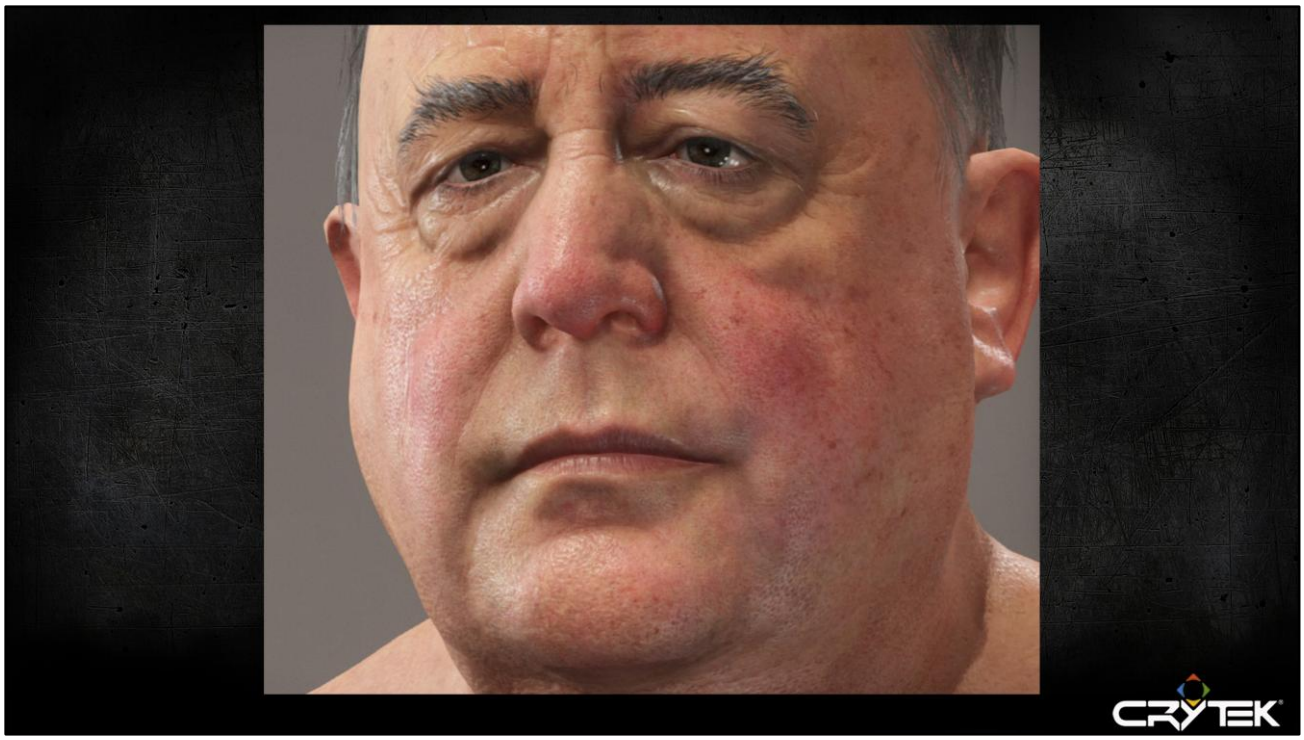

In the case of Marius the heads topology would be built from scratch on top of the scan. This mesh would later be used for all other heads as a base. Having the same topology is pretty much standard, making it easy to share information between different assets.

Once the heads are aligned they would be brought in to Zbrush. Projecting the scan data upon the new clean mesh.

The first thing an artist would have to do then is cleaning up all the scanning artefacts, hair and cloth, before moving on to the last step.


In the final step of the high poly is where the fun begins. A bit of stylization could be applied, enhancing certain features, but we made sure to keep mouth and eyes as close as possible to the actors, so the facial animations would work as good as possible.

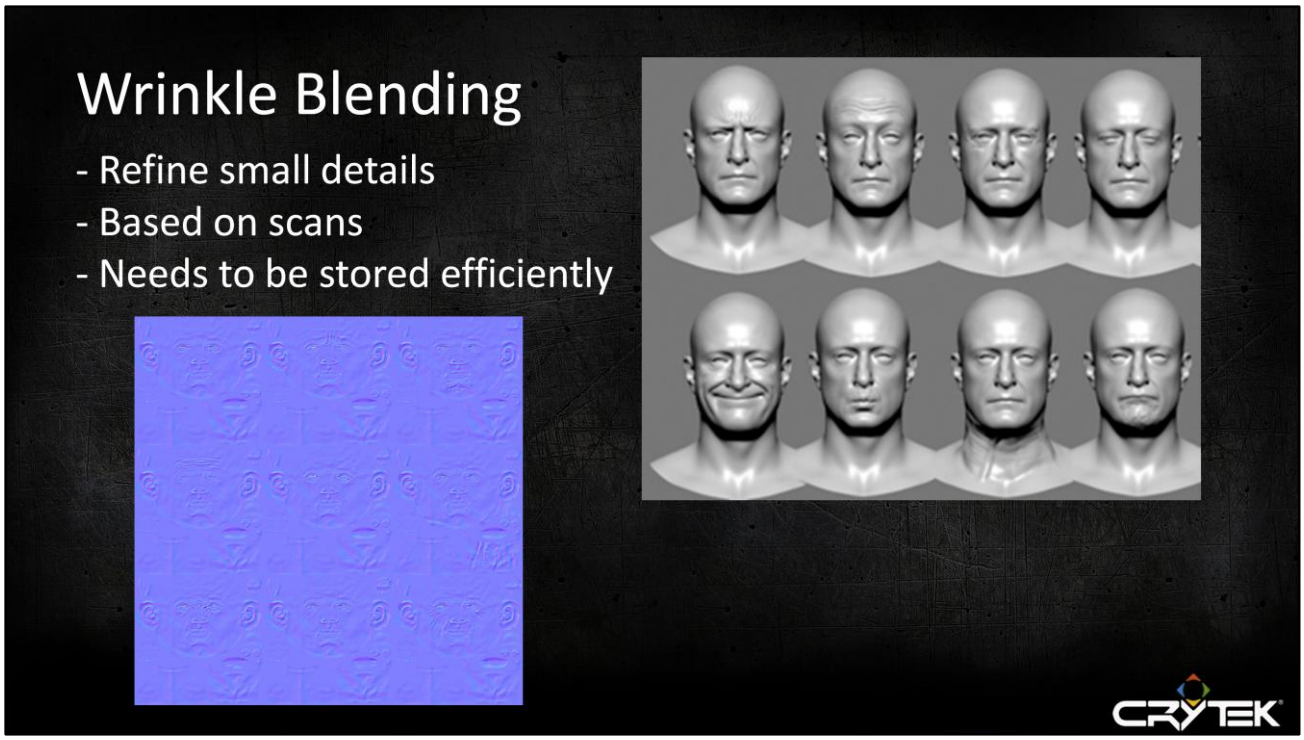
Skin details like pores wrinkles and scars would be added as well, and the end result is a finished head sculpt.




Textures were mostly hand painted for the faces, the exception would be if we had really tight deadlines, then we would use diffuse information from the scan data.

The scan data's textures still came in handy though, being used as hue reference for the artists when painting.

We also baked down small details like pores and scars from the high poly head and used these as masks in the creation of the final texture.

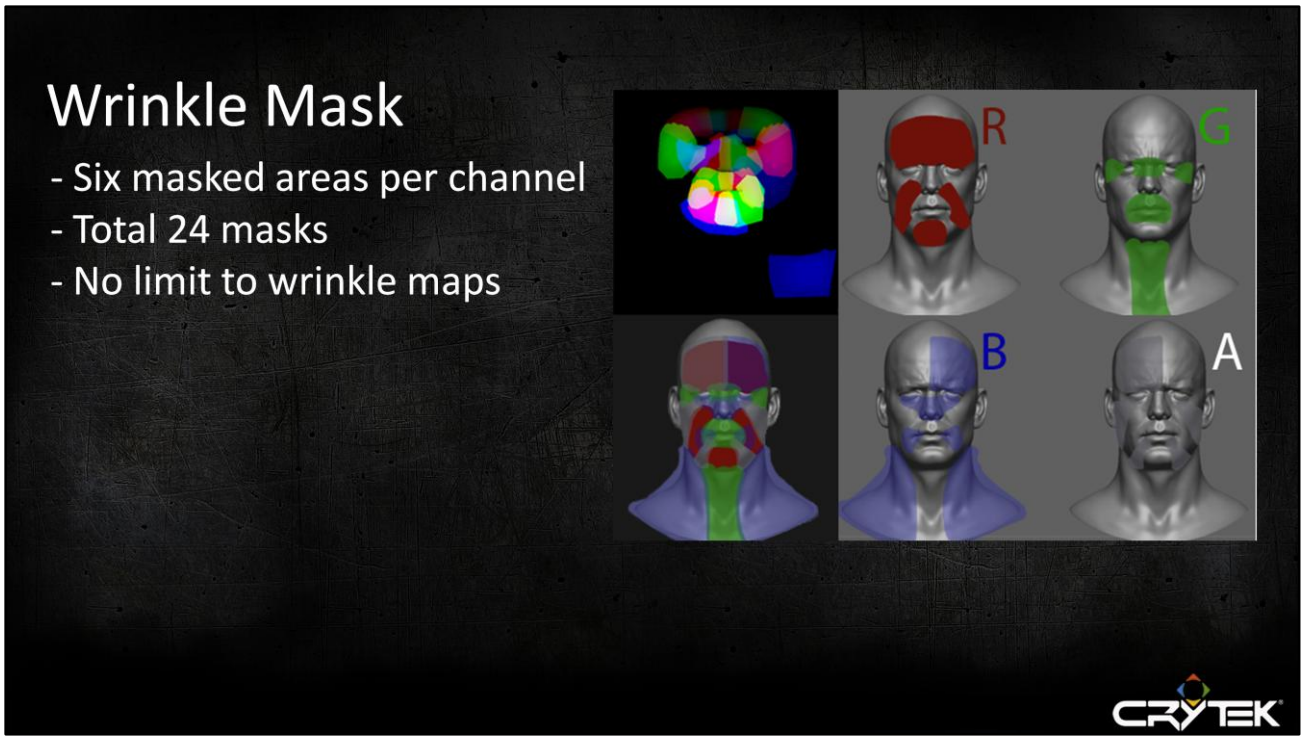



The facial rigs bones and blend shapes did an excellent job portraying the actors performance
To enhance the more subtle skin deformation normalmap blending would be needed, or wrinkle maps as we call them.

These maps will be driven by the rig and add more subtle details to the performance.

Baked down from scans of the actor making various facial expressions

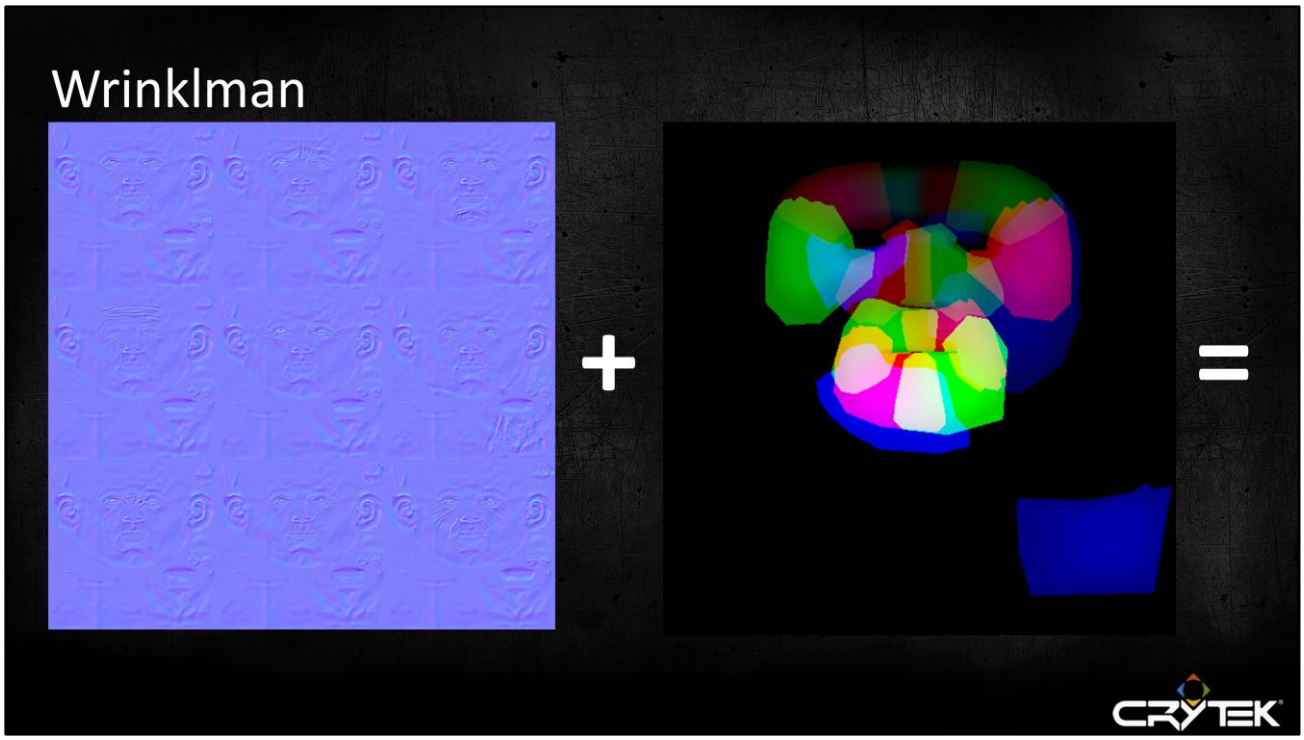
As you can see putting all the normalmaps next to each other ends up with a lot of redundant data. For example if there is wrinkles on the forehead you don't need the normalmap information from the back of the skull. So this needs to be stored in a more efficient way.




The blending system we built relied on a few different things, the first one being the Wrinkle Mask.

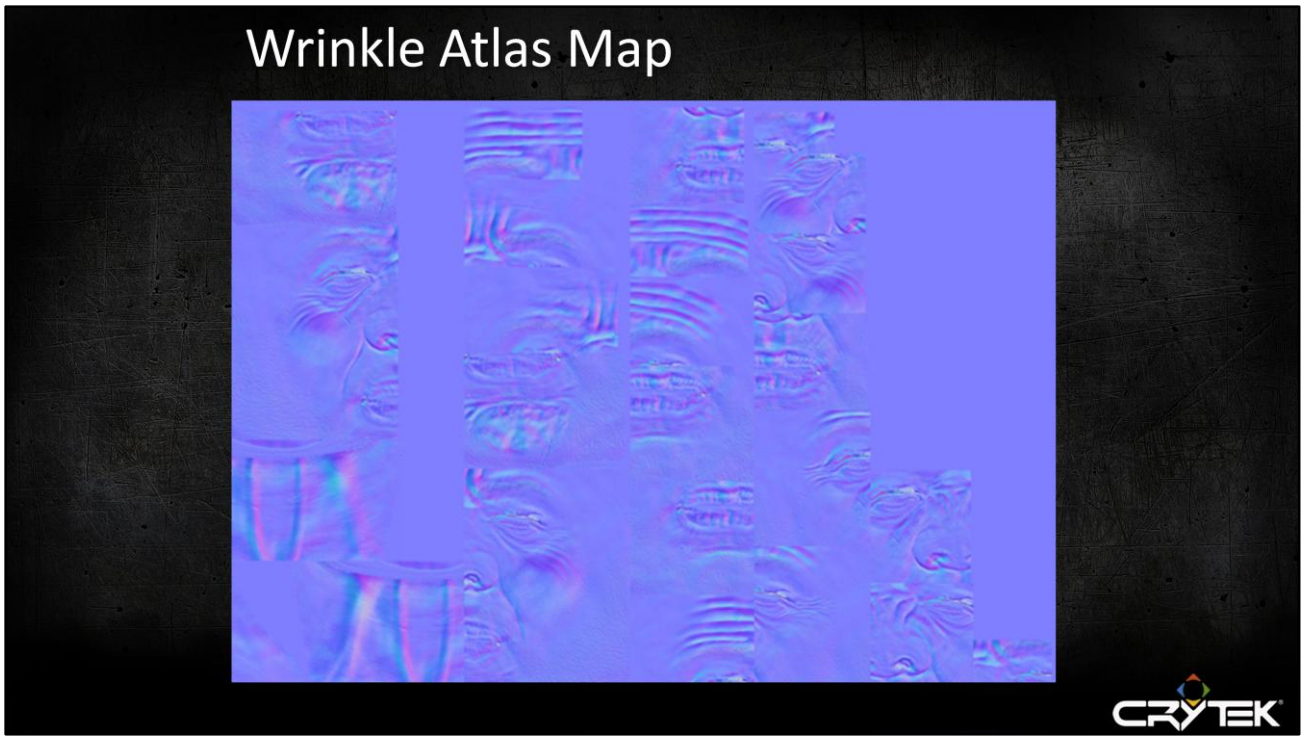
This map defines where the different wrinkle normals should blend in. Each of the textures channels can contain 6 masks, resulting in a total of 24.

But each of these masks can have multiple wrinkle normals linked to it, making it possible to blend a ton of different wrinkles to the face.




An in house tool was created called Wrinkle Man.

When Wrinkle Man is feed the normalmaps and the Wrinkle Maps the end result will be a more optemized wrinkle atlas (see the next slide).




As you can see the wrinkle atlas contains less redundant data, and as a result the wrinkles get more texture resolution.

When the Wrinkle Atlas Map and the Wrinkle Mask is hocked up to the shader the wrinkle blending can begin.

There is also an external xml file containing the coordinates for how the wrinkles are connected to the masks.


Original presentation had an video of the wrinkle blending in action, here is instead two images portraying the effect.

As you can see, with the wrinkle blending the foreheads folds are much more defined.

# SPECIAL THANKS!

## TECHNICAL ART

Riham Toulan
Sascha Herfort
Harald Zlattinger
Alex Raab
Franco Bresciani

## PROGRAMMING

Nicolas Schulz
Axel Gneiting
Bogdan Coroi
Ivo Herzeg
Andy Rayson

CRYTEK